Outline: Illumination Models

- What is light?
- Simple illumination models.
- Specular reflection.
- Shading polygons and meshes.
- Rendering pipelines.

What is Light?

- Particle or wave??
- Consider it a wave:
  - wavelength: colour;
  - amplitude: intensity.
- The visible spectrum ranges from 400 nm (violet) to 700 nm (red).
- The way we perceive light is determined by our physiology.

Light and Objects

- When light hits a surface it may either be:
  - absorbed (and often remitted at another frequency),
  - reflected (bounced straight back from whence it came),
  - scattered (bounced off the object in many directions),
  - refracted (transmitted through the object, but changing the direction),
  - transmitted (passes right through the object unaltered).
- Most computer graphics solutions to lighting are empirical.

Global Illumination Methods

- Global illumination methods try and account, in a physically based manner, for the interchange of light between all surfaces and light sources in the application model.
- Recursive ray tracing, traces rays from the viewer into each pixel and then uses reflection, refraction etc. to compute the pixel colour. If the viewer moves the whole thing must be recomputed.
- Radiosity methods compute an equilibrium lighting using a physical model. Radiosity methods allow the objects to be viewed from any angle but take an enormous amount of time to compute.
- We consider how to simulate some of these effects cheaply.

Illumination Models

- Simplest form of illumination is to give each object an illumination property of its own. We could write this model as:

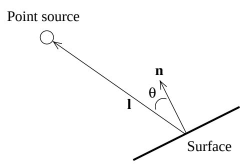
$$I = k_i \ ,$$

  where I is the intensity of the observed light, and k_i is the i^{th} objects intrinsic emission intensity.
- This is not very realistic.
- If we assume a light source (which is diffuse) then:

$$I = I_a k_a \ ,$$

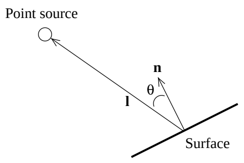
  where I_a is the intensity of the ambient light, and k_a is the objects ambient reflection coefficient.

Illumination Models



- Consider a light source that is a point from which light emanates uniformly.
- The brightness will vary with the distance to the object (since the area illuminated will increase).
- Matt surfaces will produce diffuse reflection (Lambertain reflection).

Illumination Models



- The intensity only depends on the angle between the light source and surface normal:

$$I = I_p k_d \cos(\theta) \ ,$$

  where I_p is the intensity of the point light source and k_d is the objects diffuse reflection coefficient.

Illumination Models

- The intensity of the light does not depend on the viewer angle. Note we could also write:

$$I = I_p k_d (\boldsymbol{n} \cdot \boldsymbol{l}) \ .$$

- The lighting equation must be computed in world coordinates (prior to any projections).
- A directional light source (such as the sun) can be represented by a point at infinity.
- Combining the effect of ambient lighting and diffuse reflection yields:

$$I = I_a k_a + I_p k_d (\boldsymbol{n} \cdot \boldsymbol{l}) \ .$$

Illumination Models

- If we are trying to model light sources accurately then we should account for the affects of light attenuation. Define an attenuation factor, f_{att}, to give:

$$I = I_a k_a + f_{att} I_p k_d (\boldsymbol{n} \cdot \boldsymbol{l}) \ .$$

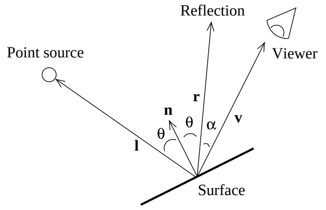
- We often use:

$$f_{att} = \frac{1}{d_l^2} \ ,$$

where d_l is the distance to the light source.

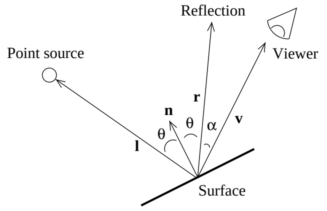
- If we include the effect of the distance of the object from the viewer we can model atmospheric attenuation to produce depth cueing. We can use the z coordinate in the projected canonical view volume.

Specular Reflection



- The shiny spots on objects.
- They are composed of reflected light, thus are the colour of the incident light.

Specular Reflection



- Specular reflection varies with the viewing angle.
- Phong illumination model, applies to imperfect mirrors.
- Reflection not just at the reflection vector $\boldsymbol{r}$, but also within a small angle, α, about that reflection vector.

Specular Reflection

- The Phong model says that the amount of light reflected is equal to $\cos^n(\alpha)$.
- This gives:

$$I = I_a k_a + f_{att} I_p \left(k_d (\boldsymbol{n} \cdot \boldsymbol{l}) + k_s (\boldsymbol{r} \cdot \boldsymbol{v})^n \right) \ ,$$

where k_s, the specular reflection coefficient is generally set using aesthetics (to produce pleasing results). Choosing the power n is also arbitrary and can be set between 3 and 200.

- Coping with multiple light sources is relatively simple, we simply sum the diffuse and specular terms for each light source.
- Colour is accounted for by having a spectral dependency in the reflection coefficients.

Lighting in OpenGL

- For each (area) object we must set the normal – either for each vertex, or once for the whole polygon (note the normal points out of the object, so the vertices must be given in anti-clockwise order).
- Each (area) object must then be given certain material properties using `glMaterial#`, which correspond to the models discussed earlier.
- These properties can be different on the front and back faces – this is an argument in `glMaterial#`.
- Then we need to specify the lights: positions and properties and enable them.

Material properties in OpenGL

- `GL_DIFFUSE` – the diffuse reflection coefficient, [0 - 1], for each R, G, B component.
- `GL_AMBIENT` – the ambient reflection coefficient, as above.
- `GL_SPECULAR` – the specular reflection coefficient, as above.
- `GL_EMISSION` – the intrinsic emission intensity, [0 - 1], for each R, G, B component.
- `GL_SHININESS` – the specular exponent.
- All these can have alpha values too.

Light properties in OpenGL

- Can define many light sources in OpenGL. Use the `glLight#` command and pass in which light (`GL_LIGHT0` to `GL_LIGHT7` and the following options:
  - `GL_POSITION` – where the light is in world coordinates.
  - `GL_DIFFUSE` – the diffuse light emitted [0 - 1] for each R, G, B component.
  - `GL_SPECULAR` – the specular light level emitted [0 - 1] for each R, G, B component.
  - `GL_AMBIENT` – the ambient light level emitted [0 - 1] for each R, G, B component.
- Note ambient light usually global, set using `glLightModel#(GL_LIGHT_MODEL_AMBIENT,value)`.

Lighting in OpenGL

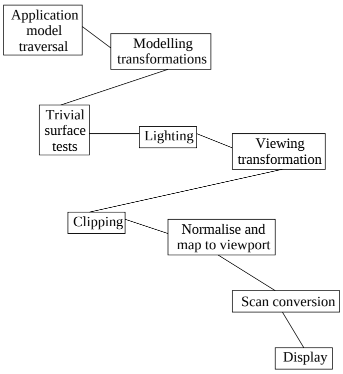
- Once the material properties are set and the lights defined (often done in the initialisation function) OpenGL must be told to switch on lighting using `glEnable(GL_LIGHTING)`.
- Each light must also be turned on using `glEnable(GL_LIGHT0)` etc.
- OpenGL also can use spotlights (we do not cover this) and implement attenuation – easy to set and add realism, but again not covered.
- Lights do not have to be static – they can be moved as part of the animation.

Shading Polygon Meshes

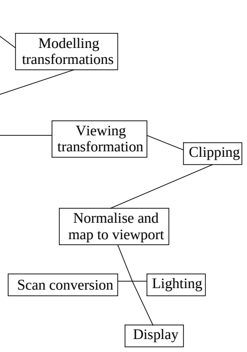
- Simplest method shades the polygon uniformly giving the flat shading model – `glShadeModel(GL_FLAT)`.
- Using interpolated shading the shading looks more realistic.
- Gouraud shading extends interpolated shading to interpolate across neighbouring polygons – `glShadeModel(GL_SMOOTH)`.
- A normal is computed at each vertex of the polygon mesh and the shading computed and then interpolated.
- Phong shading interpolates the surface normals rather than the intensities.

Rendering Pipelines

- Rendering pipelines define the complete set of processes needed to convert the data in the application model to the commands to drive the display device.
- The whole course could be characterised as exploring all parts of the rendering pipeline.



The rendering pipeline that would be used with z-buffer hidden surface removal and Gouraud shading.



The rendering pipeline, implemented for Phong shading also using the z-buffer algorithm.

Rendering Pipelines

- With Gouraud shading:
  - The illumination is computed at each vertex in world coordinates.
  - The rasterisation step (based on the z-buffer algorithm) includes the implementation of scan conversion, the interpolation of the vertex intensities and possibly any depth cueing.

Rendering Pipelines

- With Phong shading:
  - Lighting is based on interpolated normals - thus we need to be able to map the vertices (and normals) back into world coordinate space
  - We have a more expensive scheme to implement.

Summary

- Having finished this lecture you should:
  - understand lighting in computer graphics;
  - be able to contrast ambient, diffuse and specular reflection;
  - be able to implement lighting in OpenGL;
  - be able to analyse a rendering pipeline and describe the parts.
- This should start to bring the course together.